Design Credit

Active Noise Cancellation

Submitted by :-

Rhythm Jain (B20EE094)

Shivank Maurya (B20CH040)

Under the Supervision of

Dr. Amrita Puri

(Department of Mechanical Engineering)

Design/Practical Experience [EEN1010]

Department of Electrical Engineering

Academic Year: 2021-2022

Semester: 2

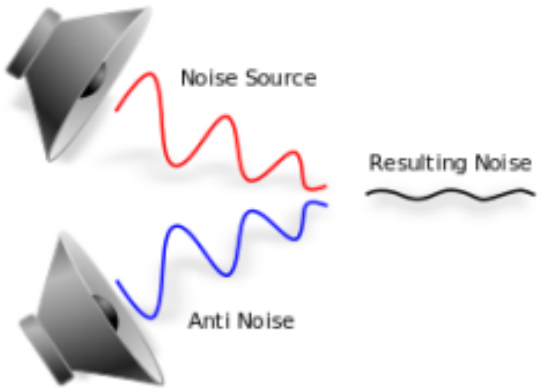
Date of Submission of Report:

1. **Name of the Student: RHYTHM JAIN**
2. **Roll Number: B20EE094**
3. **Title of the Project: Active Noise Cancellation**
4. **Project Category: 3**
5. **Targeted Deliverables:**

6. **Work Done:** **Mentioned Below**

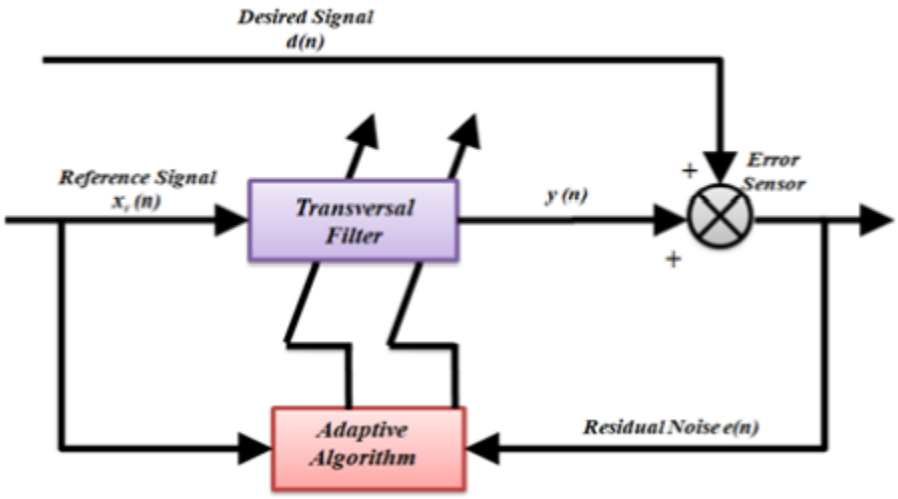
1. Introduction

Active Noise Cancellation is a technique that cancels the primary noise by generating and combining it with an anti-noise. It is based on the principle of superposition where the anti noise generated (same frequency and amplitude as that of primary noise) is 180° out of phase and is superimposed on the primary noise to create a zone of silence.



2. Methodology

Here we have worked upon various adaptive filter algorithms for ANC systems.



Block Diagram of ANC with a transversal filter.

In the above block diagram, the reference signal $x_r(n)$ is passed through the transversal filter to get the y(n) output which destructively interferes with the noise d(n) to give the error at the error sensor. This error is then used to update the weights of the adaptive filter used in the ANC system. There are various algorithms to update these weights efficiently, some of which are LMS (Least Mean Square)

algorithm, FxLMS (Filtered-X Least Mean Square), FuLMS (Filtered-X Least Mean Square) and Bilinear FxLMS.

The input signal vector to the filter is given as :

$x_r(n) = [x(n), x(n-1), \cdots, x(n-N+1)]^T$.

The weights of the adaptive filters is taken as:

$w(n) = [w_0(n), w_1(n), \cdots, w_{N-1}(n)]^T$.

Now, the output of the filter is:

$$y(n) = \sum_{i=1}^{N} x_r(n-i) * w_i(n).$$

Now, the error we get is: $e(n) = d(n) - y(n)$.

a. LMS Algorithm

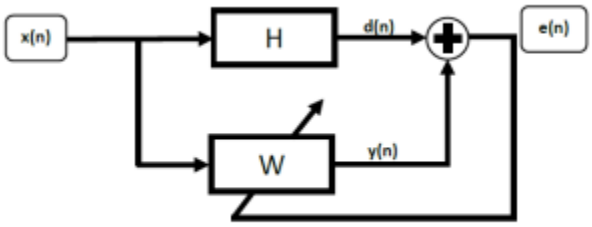


Figure 2: Block diagram of LMS algorithm [7].

The cost function we take here is J(n) given by:

$$J(n)= e^2(n).$$

The gradient of the cost function: $\Delta J(n)= 2\mu e(n)x(n)$.

Therefore the weight update equation for the LMS Algorithm is obtained as:

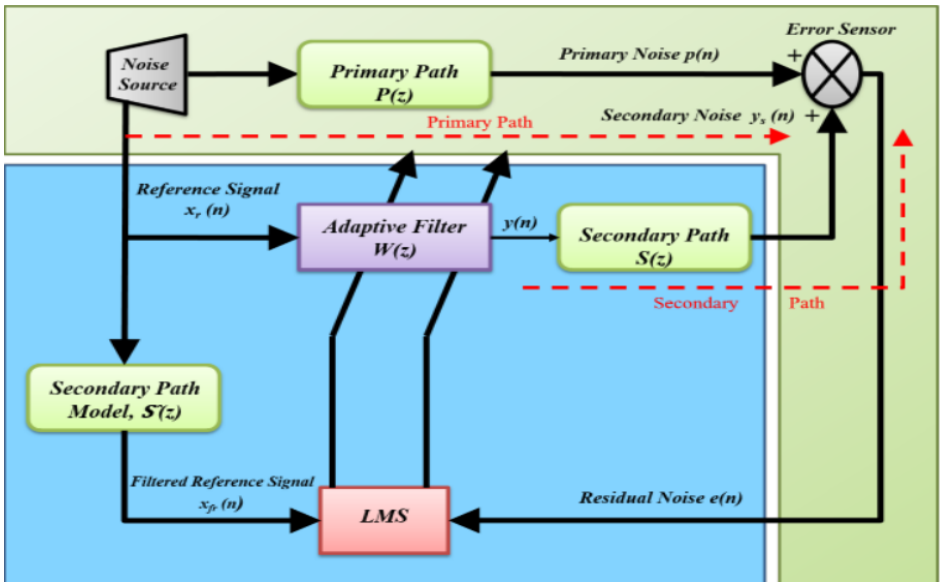
$$w(n+1)= w(n) + \mu e(n)x(n)$$

Here, μ is the learning rate or the step size (always positive) that is used to control the rate of update of the filter weights. Higher values of μ leads to divergence of error instead of convergence to 0 and much smaller value of μ leads to very slow convergence. Therefore we need to choose an optimal value of μ which we found out by trial and error method.

b. FxLMS

In this algorithm, we consider two paths namely primary path, which is from the reference sensor to the error sensor location and secondary path, which is from output of the secondary speaker (control source) to the output of the location of the error sensor.

The block diagram for the FxLMS algorithm is shown below.



Here, P(z) and S(z) denote primary path impulse response and secondary path impulse response respectively.

Here, $\Delta J(n) = 2\mu e(n)[x(n)*s(n)]$,

where s(n) is the secondary path impulse response and * denotes linear convolution.

The weight update equation is:

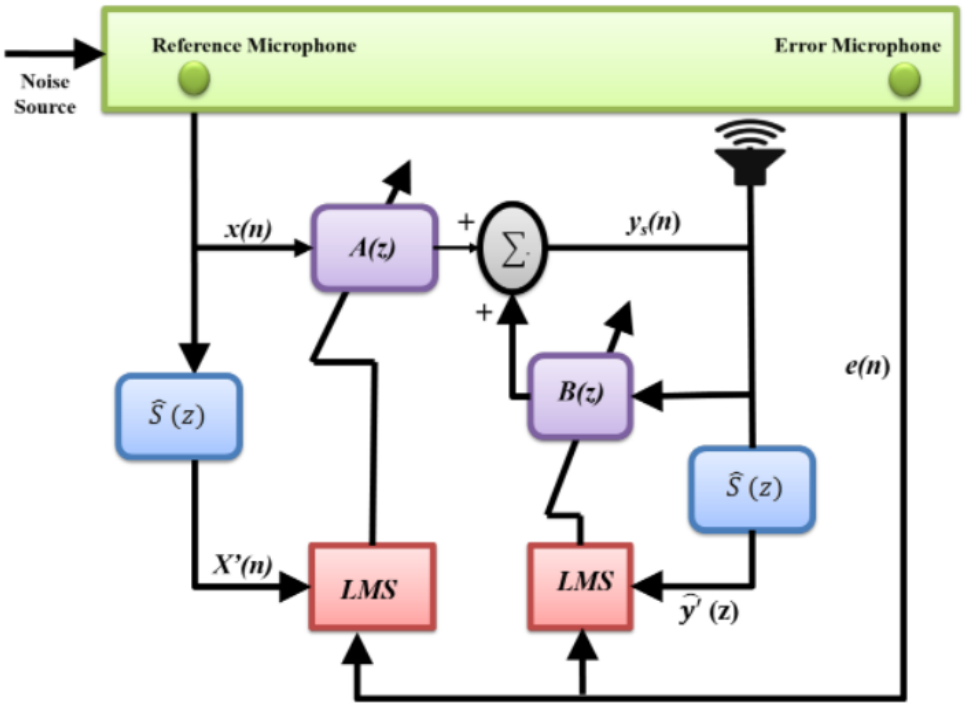
$$w(n+1) = w(n) - (\mu \div 2)e(n)x_{fr}(n)$$

$$w(n+1) = w(n) - \mu e(n)x_{fr}(n)$$

Here $x_{fr} = \widehat{s(n)} * x(n)$, that is the linear convolution of secondary path impulse response and reference signal.

c. FuLMS

The FuLMS algorithm uses adaptive IIR filters for ANC.



Block Diagram of FuLMS Algorithm

Here, an additional adaptive FIR filter B(z) is used to minimize the Mean Squared Error. The convergence rate of IIR filters is slower as compared to FIR filters and it may also not reduce the error after every iteration as the error function might converge to a local minima.

The output of the controller in FuLMS, that is :-

$$y_s(n)= \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{j=1}^{M} b_j(n)y(n-j).$$

Here, $a_i(n)$ and $b_j(n)$ denote the weight vectors of filters A(z) and B(z) respectively and, N and M represent the size of filters A(z) and B(z) respectively.

The error function of FuLMS algorithm is -

$$e(n) = p(n) + s(n) * y_s(n),$$

where s(n) is the impulse response of the secondary path.

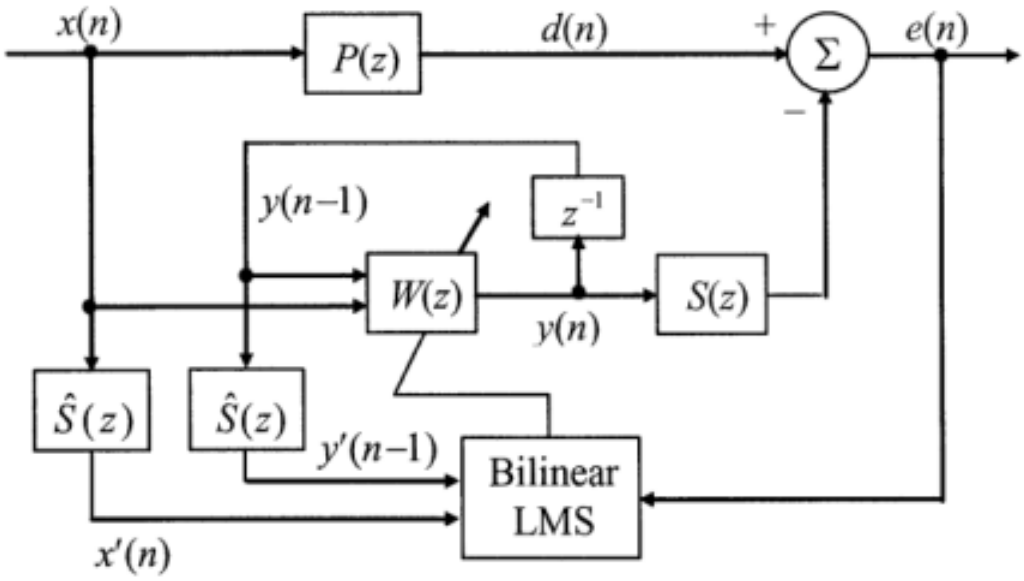
Now, the weight update equations for the FuLMS algorithm is -

$$a(n+1) = a(n) - \mu e(n)x'(n).$$

$$b(n+1) = b(n) - \mu e(n)\widehat{y'}(n-1).$$

d.Bilinear FxLMS

LMS and FxLMS are effective algorithms for ANC for linear and simple noise sources but fail in the case of non-linear and complex sources. This is where non-linear approaches come in. Bilinear is a non-linear approach and works effectively for most non-linear and complex noise sources too. Bilinear filters that employ both feedforward and feedback polynomials have a similar input-output equation as IIR filters and thus can model nonlinear systems accurately.



Block Diagram of Bilinear FxLMS Algorithm

The output of the bilinear filter is obtained as-

$$y(n)= \sum_{i=0}^{L} a_i(n)x(n-i) + \sum_{j=1}^{L} b_j(n)y(n-j) + \sum_{i=0}^{L}\sum_{j=1}^{L} c_{i,j}(n)x(n-i)y(n-j)$$

The nonlinearity of the bilinear filters is due the presence of the last term in the above equation that involves the product of input and output samples.

Here, $a_i(n)$, $b_j(n)$, $c_{i,j}(n)$ represent the weight vectors.

$$a(n) = [a_0 (n)\ a_1 (n)\ldots\ a_L (n)]^T .$$

$$b(n) = [b_1 (n)\ b_2 (n)\ldots\ b_L (n)]^T .$$

$$c(n) = [c_{0,1} (n)\ldots\ldots\ c_{0,L} (n) c_{1,1} (n)\ldots\ c_{1,L} (n)\ldots\ c_{L,L} (n)]^T .$$

For each time sample, the reference signal x(n) convolves with the primary path P(z) to produce the desired signal d(n) and goes to the Bilinear filter after convolving with the Secondary Path for weight updation. x(n) also goes to w(z) to produce the output signal vector. y(n) is the output signal vector which after convolving with Secondary Path and after adding with d(n) at the junction creates the error signal e(n) which travels back to the bilinear LMS to update weight. y(n) is also fed back to w(z) to produce output signal vector with one sample time delay y(n-1).

Now, the weight update equations are-

$$a(n+1) = a(n) - \mu_a e(n)x'(n).$$

$$b(n+1) = b(n) - \mu_b e(n)y'(n-1).$$

$$c(n+1) = c(n) - \mu_c e(n)v'(n).$$

x(n) and y(n-1) in w(z) helps to create an equation for v(n) which is defined as-

$$v(n) = [x(n)y(n-1).....x(n)y(n-L)x(n-1)y(n-1)...x(n-1)y(n-L)...x(n-L)y(n-L)]^T$$

x'(n) and y'(n-1) in Bilinear FxLms helps to create an equation for v'(n) which is used to update the weights c(n).

Implementation and Results

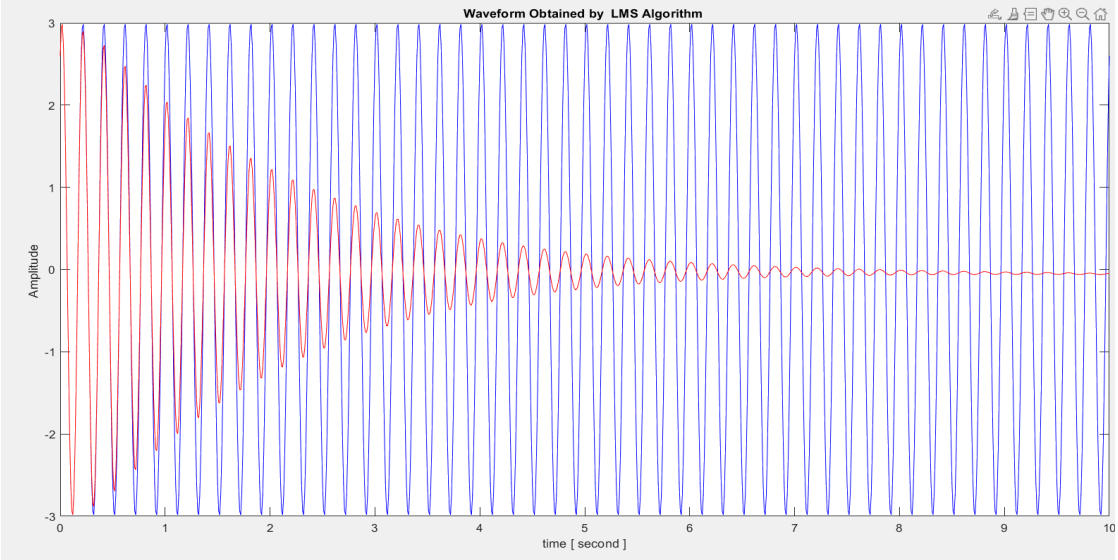
LMS(Least Mean Square)

Code:

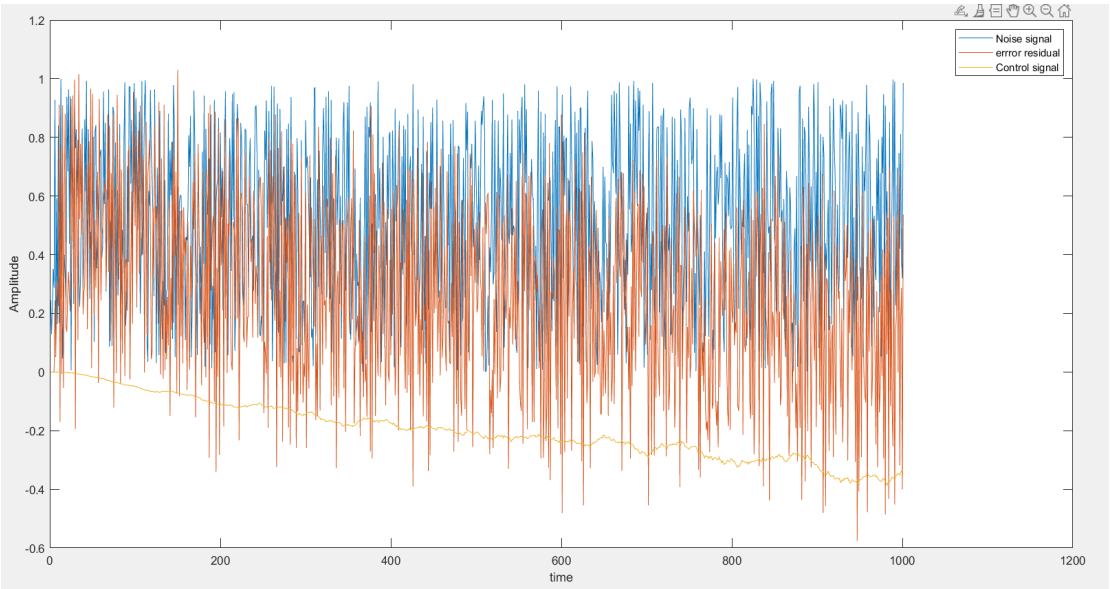
```
N=50;
W=zeros(N,1);
mu=0.0001;
Y_t=zeros();
E_t=zeros();
Xr_in=zeros(N,1);
M=1001;
T2=zeros();
for n=1:M
    Xr_in=[X_r(n);Xr_in(1:N-1)];
    Y=sum(Xr_in.*W);
    e=p(n)+Y;
    W=W-mu*e*Xr_in;
    E_t(n)=e;
    Y_t(n)=Y;
    %disp(E_t);
 n_iter=n;
end
figure(2)
plot(X_ps)
hold on
plot(E_t)
hold on
plot(Y_t)
legend('Noise signal','error residual','Control signal')
xlabel('time');
ylabel('Pressure')
```

Results.

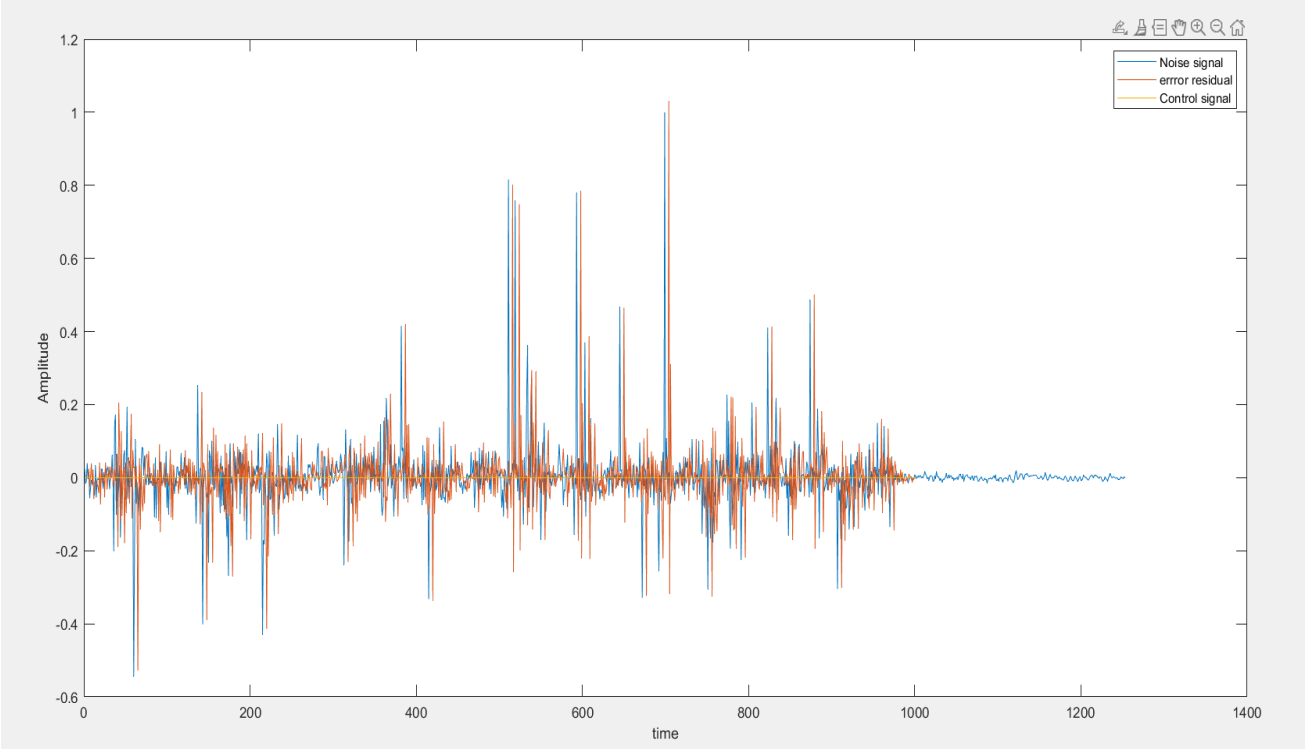
For tonal noise:



2.For Random Noise:



3.For Welding Noise



FxLMS

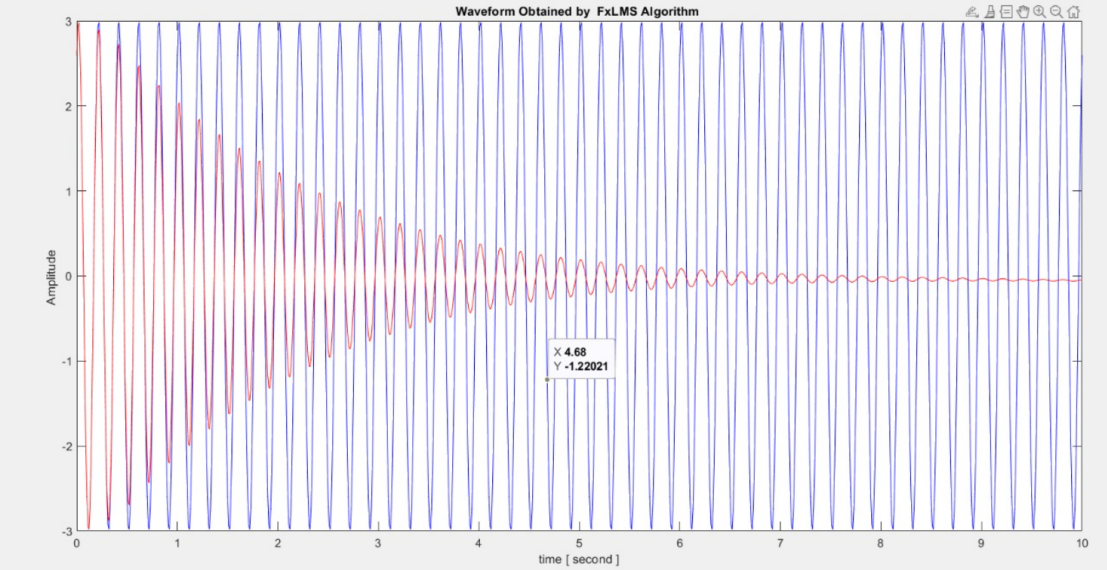
Code:

```
N=60;%filter size
W=zeros(N,1);%vector
mu=0.00001;%learning rate
Xr_in=zeros(N,1);%convolving parameter input to Fir
M=1001;
Y_t=zeros();
E_t=zeros();
Xfr=zeros(N,1);%Xfr=Xr*secd.path
Xr_2=zeros(M,1);%store all reference value over given time
%X_ps=3*sin(2*pi*freq*t+phi);
Yf=zeros(M,1);%o/p of fir
T2=zeros();
for n=1:M%convolving for entire time span using fir(n)
    Xr_in=[Xr(n);Xr_in(1:N-1)];%total size equal to fir(n)
    Y=sum(Xr_in.*W);%o/p of fir_conv sum
    Xr_2=[Xr(n);Xr_2(1:M-1)];%conv step 01
    sz=size(Y);
disp(sz); Xfr_n=sum(Xr_2.*SP);%conv step 2
 %ys=y*sec_ir
Yf=[Y;Yf(1:M-1)]; %o/p fir filter
 Ys=sum(Yf.*SP);%conv. sum
    e=p(n)+Ys;%error mic
    Xfr=[Xfr_n;Xfr(1:N-1)];
    W=W-mu*e*Xfr;%update weights
    E_t(n)=e;%error in time
    Y_t(n)=Ys;%output to the fir
     n_iter=n;
    %T2(n)=t(n);

end
```

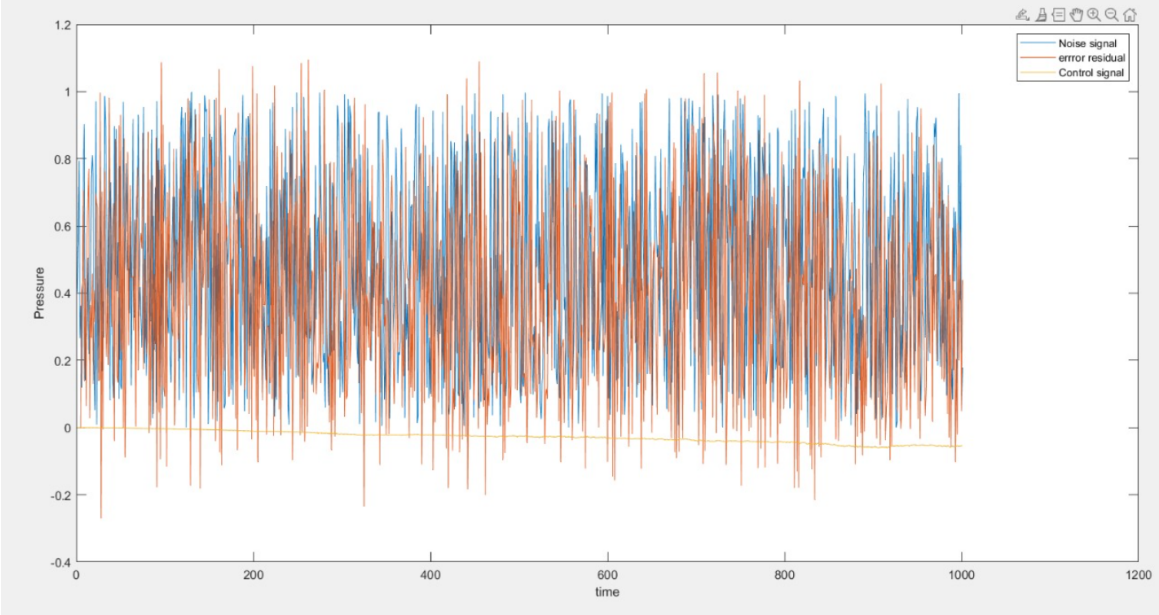
```
figure(2)


plot(X_ps)
hold on
plot(E_t)
hold on
plot(Y_t)
legend('Noise signal','error residual','Control signal')
xlabel('time');
ylabel('Pressure')
hold off;
```

Results: Error v/s Time graph for different input noises.

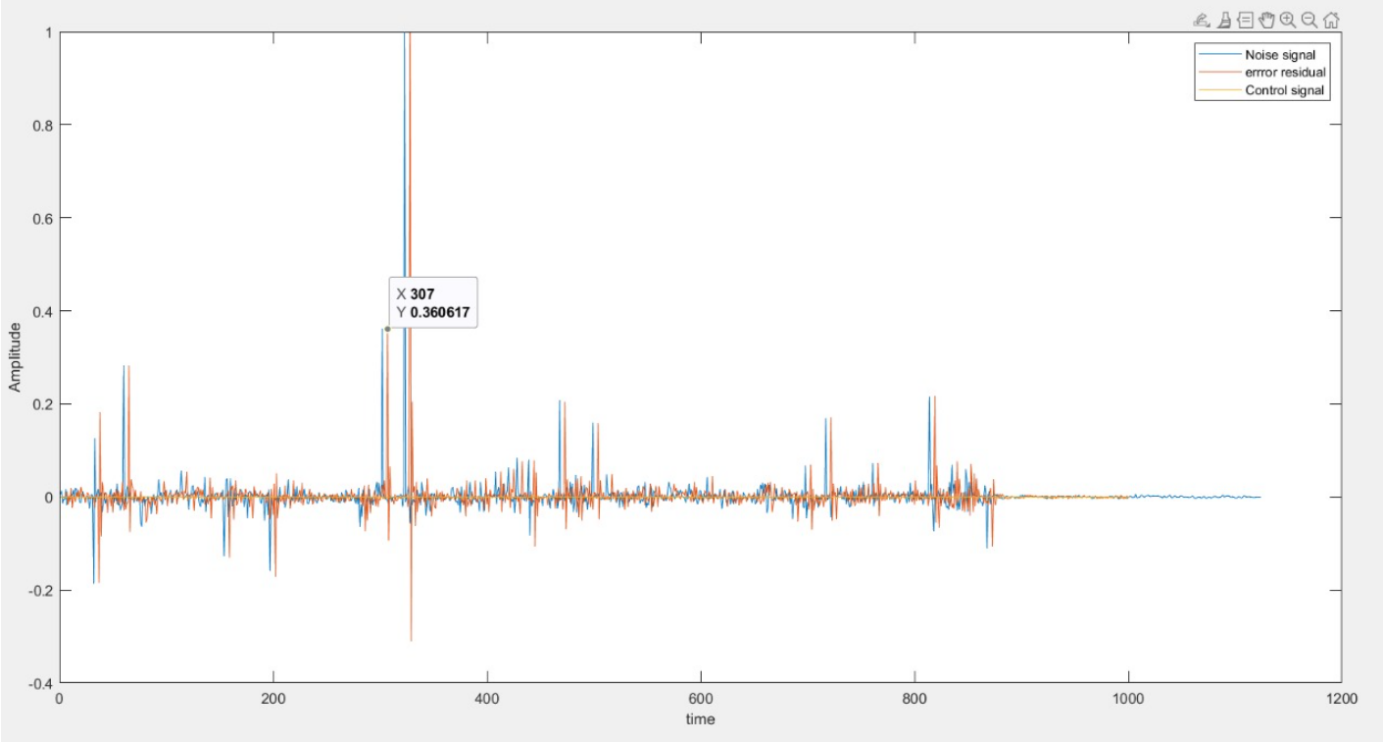
Tonal Noise-



Random Noise-



Welding Noise-



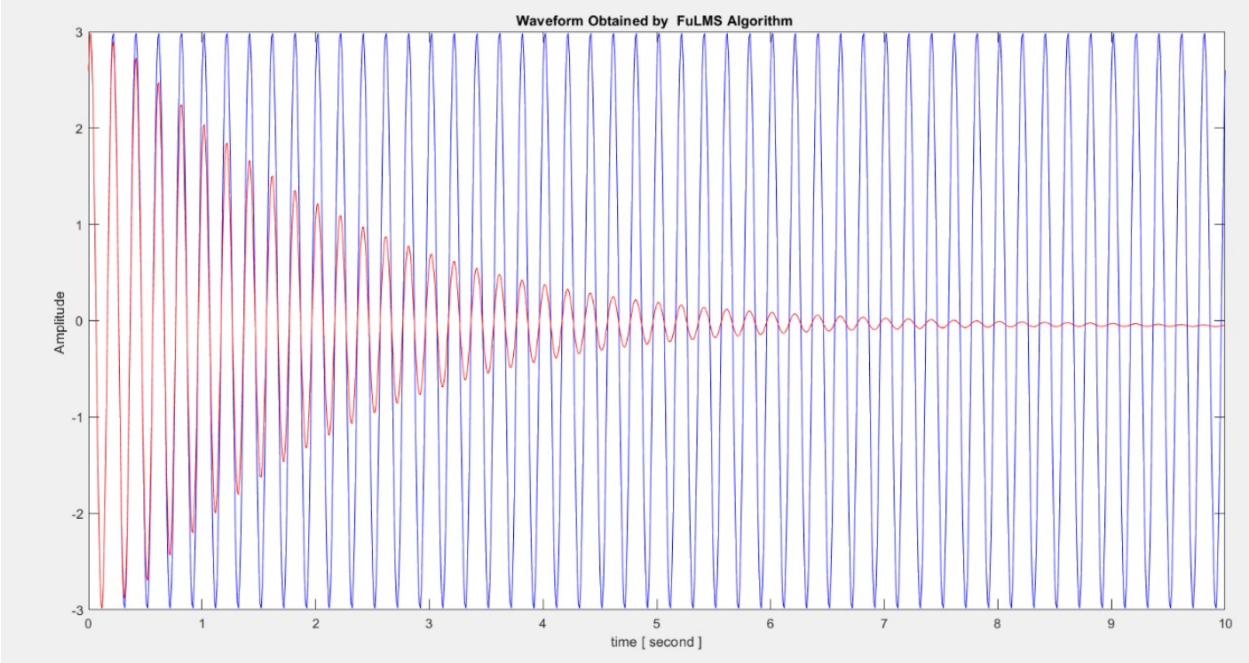
Fulms:

```
N=60;

mu=power(10,-5.1);

%M=length(t);

Aw=zeros(1,N);

Bw=zeros(1,N);

x_buf=zeros(N,1);

y_buf=zeros(N,1);

y_buf_last=zeros(N,1);

y_buf_vector_last=zeros(1,N);

r_buf=zeros(1,N);

r_buf1=zeros(1,N);

r_1=zeros(N,1);

r_2=zeros(N,1);

E_t=zeros();

y_signal=zeros(N,1);

x_buf1=zeros(N,1);

control_signal=zeros();

T2=zeros();

for n=1:M

    x_buf=[X_ps(n);x_buf(1:N-1)];%storing reference signal

    r_1=[X_ps(n);r_1(1:N-1)];

    r=sum(r_1.*SP);%convolution with secondary impulse response

    r_buf=[r,r_buf(1:end-1)];
y_sig=sum(Aw*x_buf(1:N))+sum(Bw*y_buf_last);

    %disp(y_sig);

    y_signal=[y_sig;y_signal(1:end-1)];

    %sz1=size(y_sig);

    %disp(sz1);

    y_buf=[y_sig;y_buf(1:N-1)];

    r1=sum(y_buf.*SP);

  %disp(r1);

    r_buf1=[r1,r_buf1(1:end-1)];

    e=p(n)+r1;

  Aw=Aw-mu*e*r_buf;

  Bw=Bw-mu*e*y_buf_vector_last;

  y_buf_last=y_buf(1:N);

  y_buf_vector_last=r_buf1(1:N);

  E_t(n)=e;

  %sz1=size(r_buf1);

 %disp(r_buf1);
```

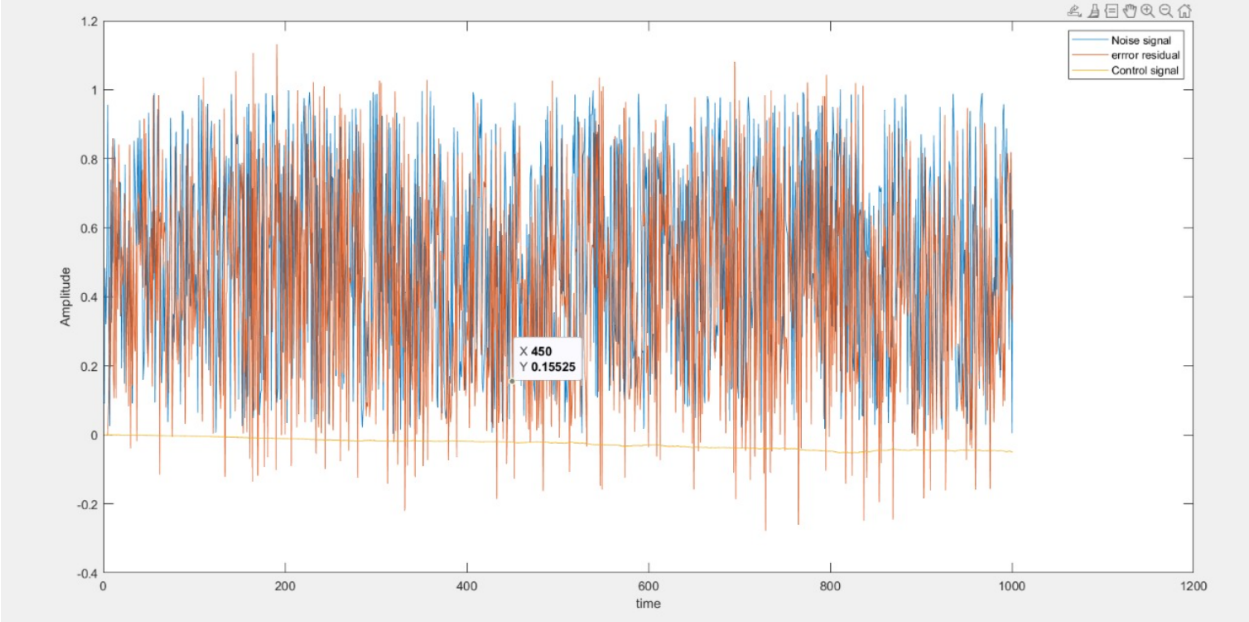
```
    %r1=sum(r_2*Y_sec_lr);
    plot(y_signal,'*--');
    n_iter=n;
    control_signal(n)=y_sig;
    %T2(n)=t(n);
end
figure(2);
plot(X_ps)
hold on
plot(E_t)
hold on
plot(control_signal)
legend('Noise signal','error residual','Control signal')
ylabel('Amplitude');
xlabel('time');
hold off;
```

Results: Error v/s Time graph for different input noises.

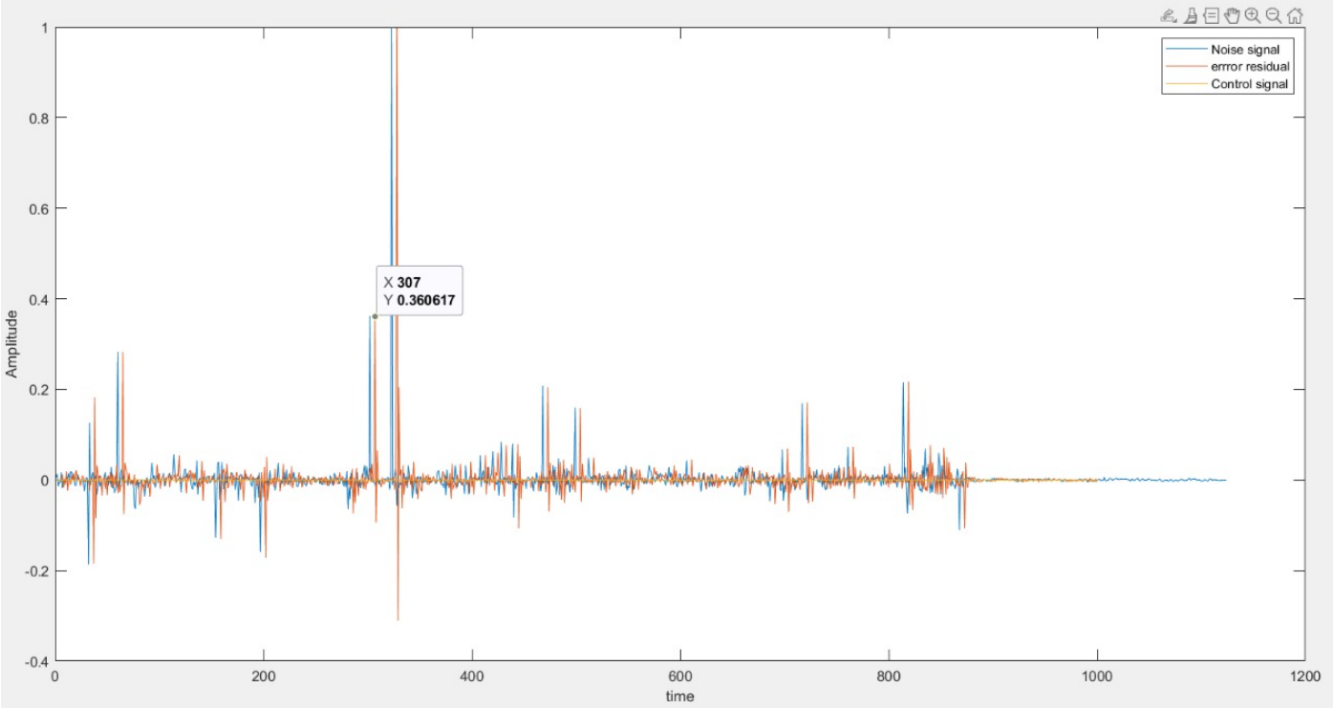
Tonal Noise-



Random Noise-



Welding Noise-



Bilinear Fxlms

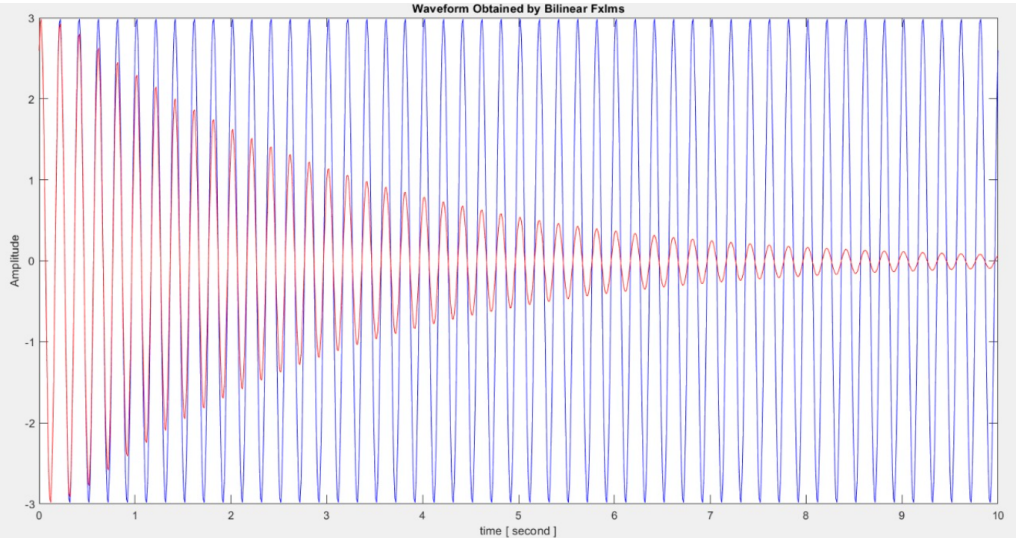
Code:

```
N=60;
mu1=1.9*power(10,-2);
mu2=1.5*power(10,-2);
mu3=power(10,-3);
Aw=zeros(1,N+1);
Bw=zeros(1,N);
Cw=zeros(N+1,N);
Cw1 = zeros(1,N^2 + N);
x_buf=zeros(N+1,1);
y_buf=zeros(N,1);
V_buf = zeros(N^2 +N,1);
y_buf_last=zeros(N,1);
y_buf_vector_last=zeros(1,N);
r_buf=zeros(1,N+1);
r_buf1=zeros(1,N);
r_1=zeros(N,1);
r_2=zeros(N,1);
E_t=zeros();
y_signal=zeros(N,1);
v_n=zeros(N,1);
v_n1=zeros(N,1);
v_buf=zeros(1,N^2+N);
x_buf1=zeros(N,1);
control_signal=zeros();
T2=zeros();
y_sig = 0;
for n=1:M
    x_buf=[X_ps(n);x_buf(1:end-1)];
%storing refernce signal
    r_1=[X_ps(n);r_1(1:N-1)];
    r=sum(r_1.*SP);
    r_buf=[r,r_buf(1:end-1)];
   V_buf = [X_ps(n)*y_sig; V_buf(1:N^2
+ N -1, 1)];
    y_sig= Aw*x_buf + Bw*y_buf_last +
Cw1*V_buf;
    y_signal=[y_sig;y_signal(1:end-1)];
    v=sum(V_buf(1:N).*SP);
    v_buf=[v, v_buf(1:end-1)];
    y_buf=[y_sig; y_buf(1:N-1)];
    r1=sum(y_buf(1:N-1).*SP(1:N-1));
```

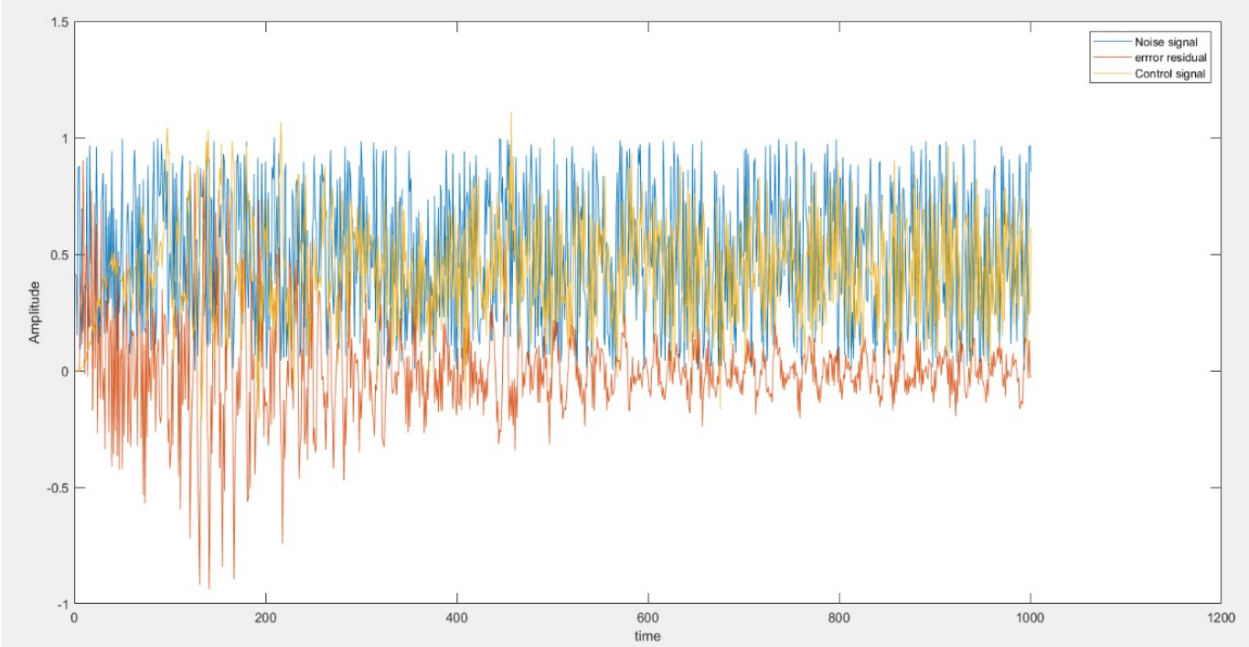
```
        r_buf1=[r1,r_buf1(1:end-1)];
        e=p(n)-r1;
    Aw=Aw+ mu1*e*r_buf;
    Bw=Bw+ mu2*e*y_buf_vector_last;
    Cw1 =Cw1 + mu3*e*v_buf;
    y_buf_last=y_buf(1:N);
    y_buf_vector_last=r_buf1(1:N);
    E_t(n)=e;
    n_iter=n;
    control_signal(n)=y_sig;
end
plot(X_ps)
hold on
plot(E_t)
hold on
plot(control_signal)
legend('Noise signal' ,'error residual','Control signal')
hold off;
ylabel('Amplitude');
xlabel('time');
function sys3 = IMPULSE1(num,den,Ti,Ts,Tf)
sys = tf(num, den, Ts);
sys3 = impulse(sys,Ti:Ts:Tf);
end
```

Results: Error v/s Time graph for different input noises.

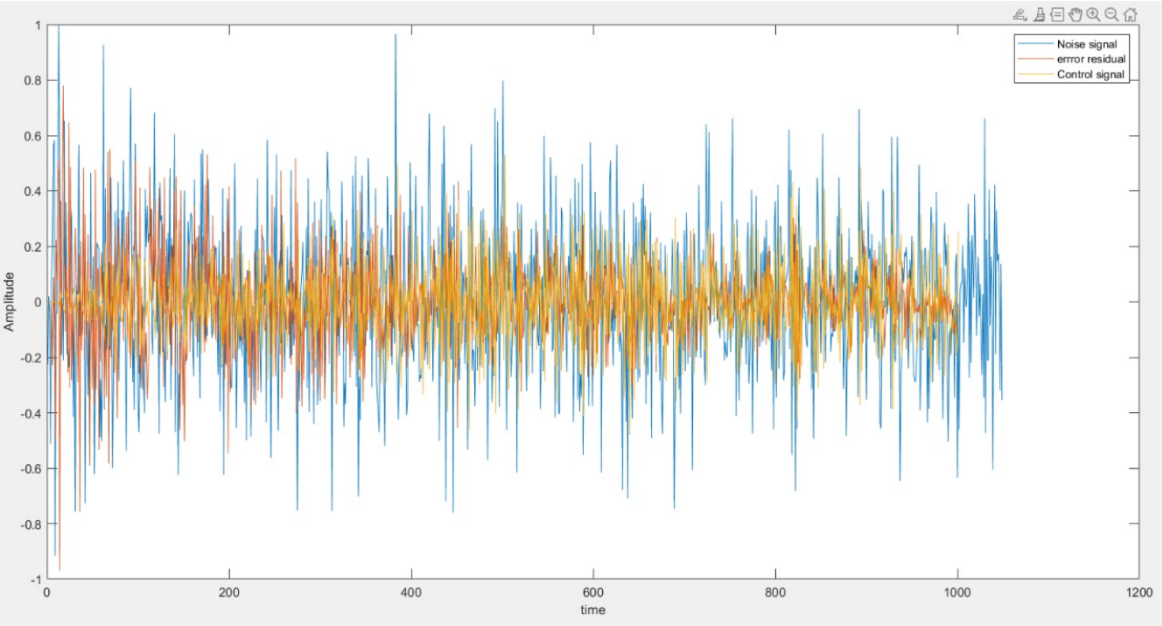
Tonal Noise-



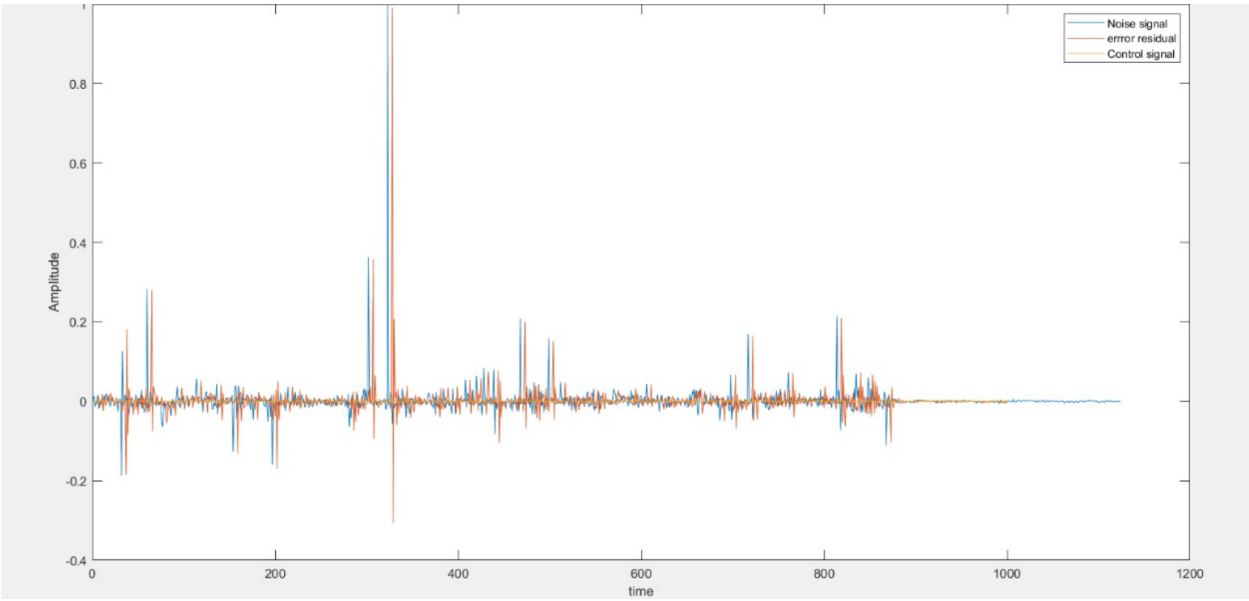
Random Noise-



Universal Mill Noise-



Welding Noise-



Conclusion

In conclusion, we studied a total of 4 Algorithms namely LMS, FxLMS, FuLMS and Bilinear FxLMS.

From the output graphs, we observe that LMS, FxLMS and FuLMS work quite well in reducing the error for linear noises but as soon as the welding noise was provided as the input signal, they failed to reduce the error. Also, out of these 3 algorithms, FxLMS is the fastest to converge.

Bilinear FxLMS worked well on the linear noises as well as on the non-linear noises like universal mill but again failed to reduce the error in the Welding noise.

References

[1] Nonlinear Adaptive Bilinear Filters for Active Noise Control Systems by Sen M. Kuo, Senior Member, IEEE, and Hsien-Tsai Wu, Member, IEEE.

[2] Active Noise Reduction using LMS and FxLMS Algorithms Krishna A/L Ravinchandra, Ka Fei, Thang and Chee Yong, Lau.

[3] Feed-Forward Active Noise Control in Vibro-Acoustic Cavities, Thesis by Vikas Kumar Lakhman.

[4] Understanding Active Noise Cancellation by Colin H Hansen.

Declaration: I declare that no part of this report is copied from other sources. All the references are properly cited in this report.

Supervisor's Recommendation for the Evaluation

Please tick any one of the following

1. The work done is satisfactory, and sufficient time has been spent by the student. The submission by the student should be evaluated in this term.
2. The work is not complete. Continuity Grade should be given to the student. The student would need to be evaluated in the next semester for the same Design Project with me.
3. The work is not satisfactory. There is no need for evaluation. The students should look for another Design Credit Project for the next semester.
4. [Other Comment, if 1-3 are not valid]

   _____

Signature of the Supervisor